THE UNIVERSITY OF
MELBOURNE

COMP 90048
Declarative Programming
Workshop 6 (week7)

2019 semester 1

by Wendy Zeng

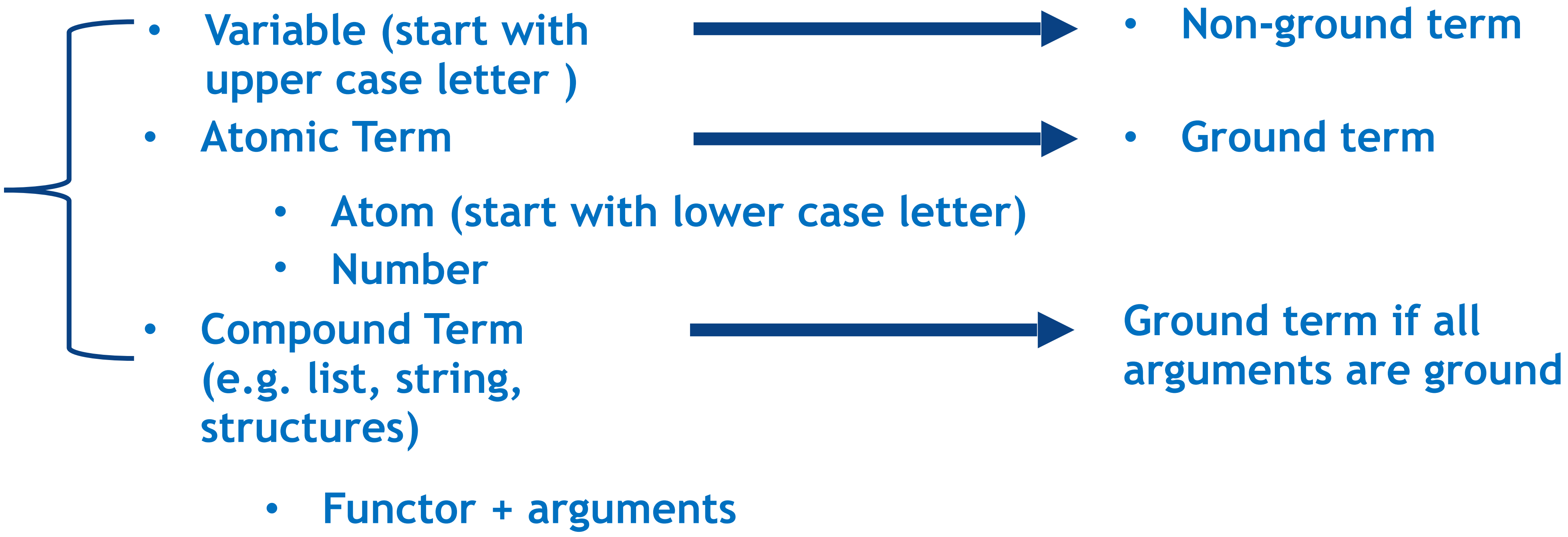
Tutorial : Tue 18:15 - 19:15 221 Bouverie St, room B11
Wed 17:15 - 18:15 201 Bouverie St, room B132

Outline

1. Prolog Types and datalog
2. Facts, Rules and Queries
3. Negation as Failure
4. Unification

1. Prolog Types and Datalog

- Porlog basic data structure: term
- Both predicates and data are expressed in the form of terms
- Types of terms:



- Variable (start with upper case letter) → Non-ground term
- Atomic Term → Ground term
  - Atom (start with lower case letter)
  - Number
- Compound Term (e.g. list, string, structures) → Ground term if all arguments are ground
  - Functor + arguments

2. Facts,Rules and Queries

- A prolog programs consists of facts and rules
- Predicate:
  - Name of the relationship between objects
- Facts:
  - Describe the relationship between objects which is always true
  - Each fact corresponds to one “entry” in the database
  - Head but no body
- Rules:
  - Describe the conditions of a fact or the relationships between facts
  - Head :- Body (:- reads as “if”)

- Example1:
  - city(bangkok,thailand,1178).
- Example2:
  - borders(france,spain).
- Example3:
  - larger(C1, C2) :-
    area(C1, Area1),
    area(C2, Area2),
    Area1 > Area2.

2. Facts,Rules and Queries

- Query:
  - Whether the goal can be satisfied according to the facts in the "database"
  - If the goal is satisfiable, then create variable bindings -> succeed
  - If the goal is not satisfiable, no bindings are created -> fail
- How prolog interpreter answers the query:
  - Sequential search for all facts
  - Instantiate variables
  - Backtrack if a goal fails

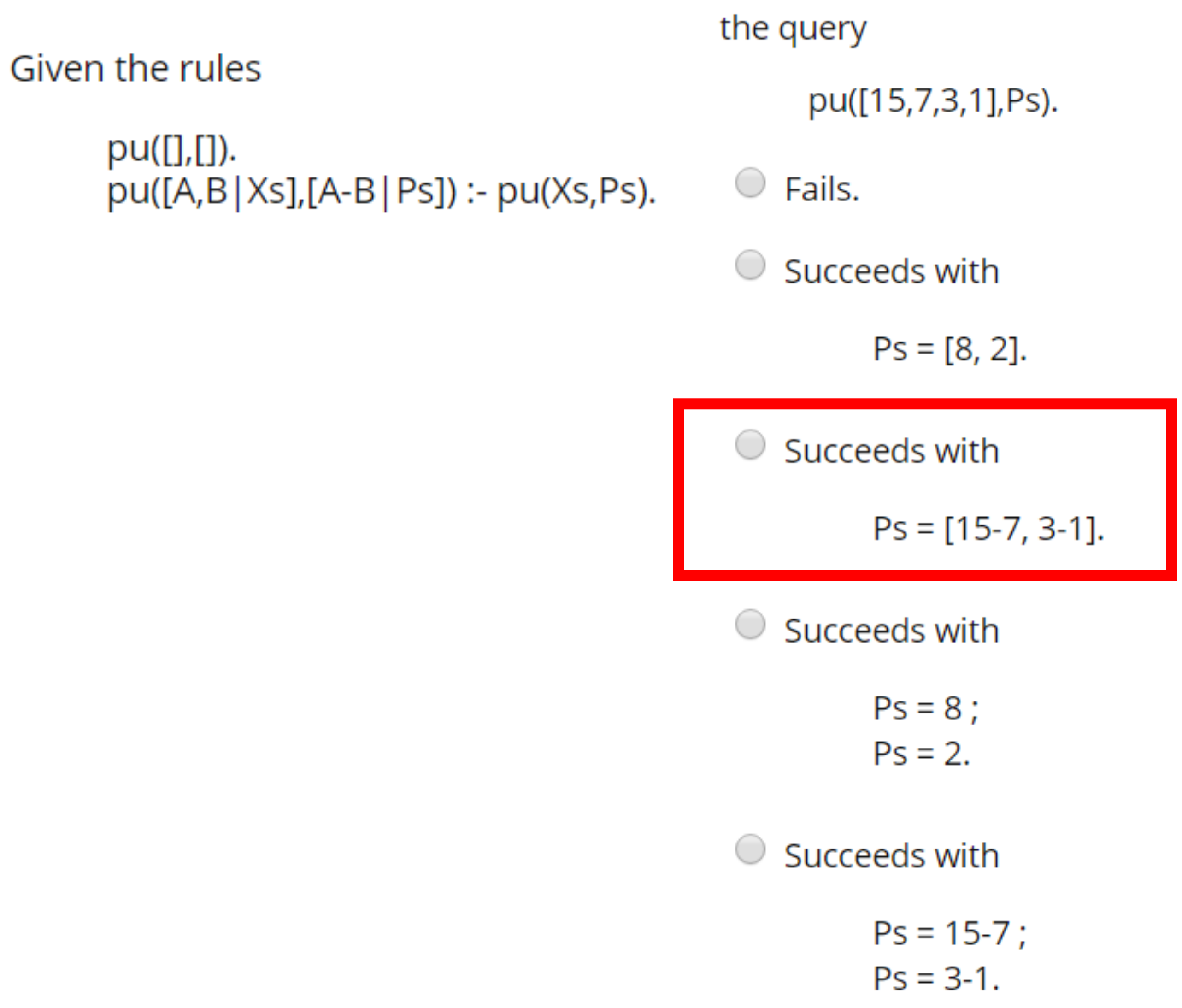
- A bit of wrapping up:
  - A predicate is described via a number of clauses
  - Each clause is either a rule or a fact
  - Use query in swipl to interactively query the statement

3.Unification

- Use the = sign:
  - Unify the right-hand-side with the left-hand-side
  - Unification can fail or succeed
  - Succeed if :
    - LHS and RHS are the same term
    - or Non-var terms are matched and var terms can be instantiated
  - Fail if terms are match, no variable bindings

- Unification rules:
  - If LHS and RHS are ground:
    - Unify iif they are identical atoms or numbers
  - If LHS is Variable and RHS is ground:
    - Unify and instantiate LHS to RHS (vice versa)
  - If LHS and RHS are complex terms:
    - They have same functor and args and all args unify
    - They have compatible variable instantiations

3. Unification



Given the rules

```
pu([],[]).
pu([A,B|Xs],[A-B|Ps]) :- pu(Xs,Ps).
```

the query

```
pu([15,7,3,1],Ps).
```

○ Fails.

○ Succeeds with

Ps = [8, 2].

[✓ ○ Succeeds with

Ps = [15-7, 3-1].]

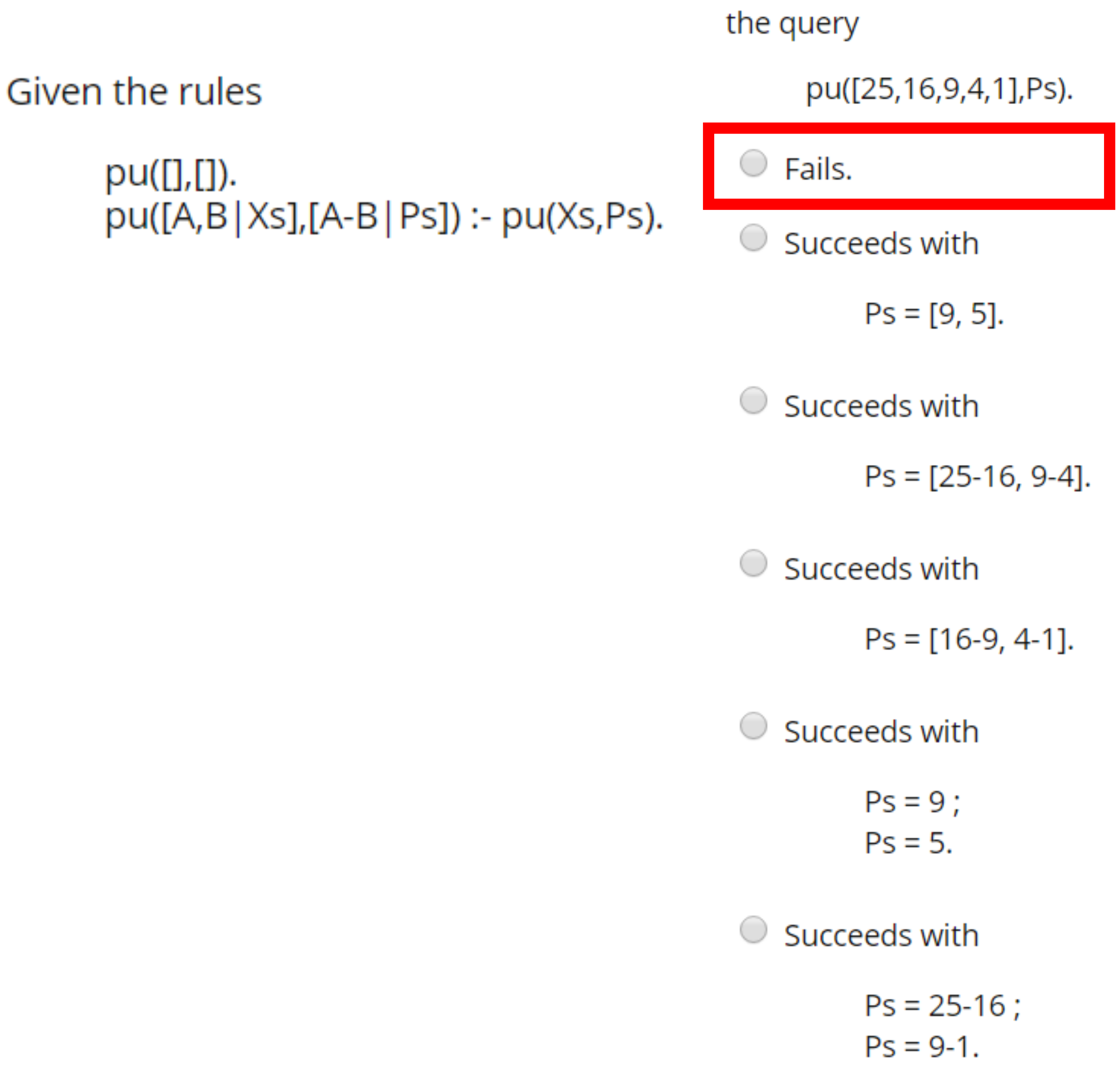
○ Succeeds with

Ps = 8 ;
Ps = 2.

○ Succeeds with

Ps = 15-7 ;
Ps = 3-1.

3. Unification



Given the rules

```
pu([],[]).
pu([A,B|Xs],[A-B|Ps]) :- pu(Xs,Ps).
```

the query

```
pu([25,16,9,4,1],Ps).
```

[✓ ○ Fails.]

○ Succeeds with

Ps = [9, 5].

○ Succeeds with

Ps = [25-16, 9-4].

○ Succeeds with

Ps = [16-9, 4-1].

○ Succeeds with

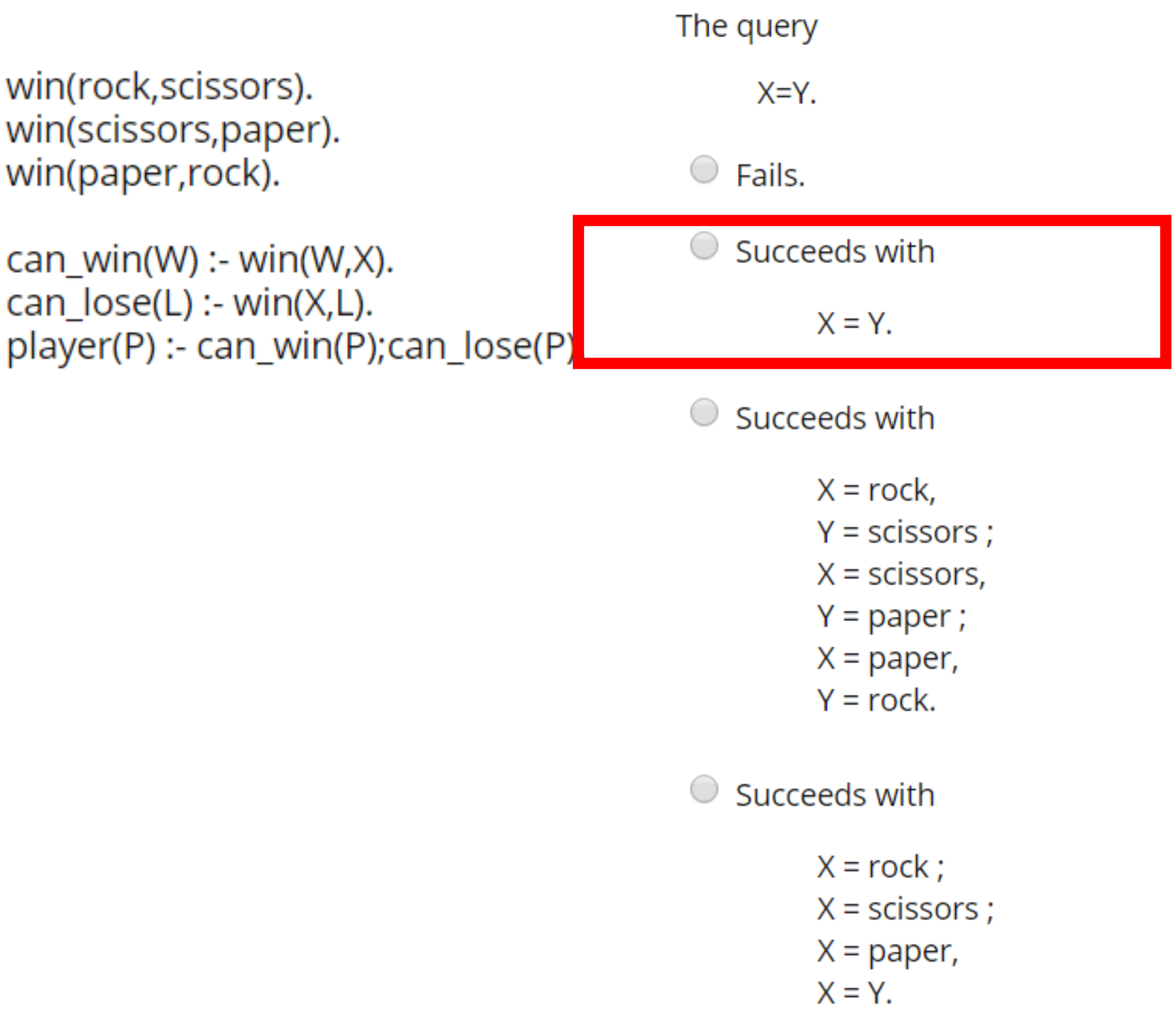
Ps = 9 ;
Ps = 5.

○ Succeeds with

Ps = 25-16 ;
Ps = 9-1.

No unifications succeed.
No pattern matching for the case where first argument is a list of one element

4. Negation as Failure



```
win(rock,scissors).
win(scissors,paper).
win(paper,rock).

can_win(W) :- win(W,X).
can_lose(L) :- win(X,L).
player(P) :- can_win(P);can_lose(P)
```

The query

X=Y.

○ Fails.

○ Succeeds with

X = Y.

○ Succeeds with

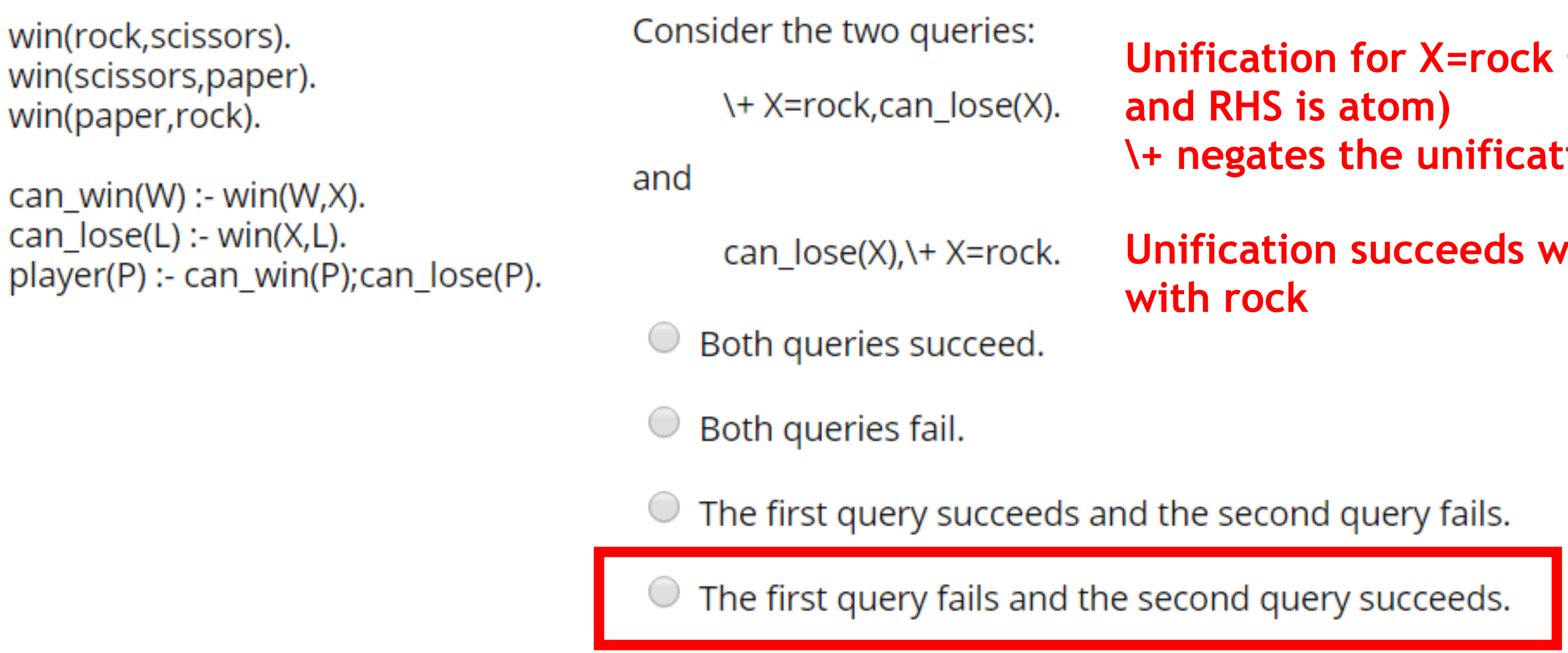
X = rock,
Y = scissors ;
X = scissors,
Y = paper ;
X = paper,
Y = rock.

○ Succeeds with

X = rock ;
X = scissors ;
X = paper,
X = Y.

LHS and RHS both Variables
Unify themselves with each other
meaning they share same values

4. Negation as Failure



```
win(rock,scissors).
win(scissors,paper).
win(paper,rock).

can_win(W) :- win(W,X).
can_lose(L) :- win(X,L).
player(P) :- can_win(P);can_lose(P).
```

Consider the two queries:

```
\+ X=rock,can_lose(X).
```

Unification for X=rock succeeds (LHS is Variable and RHS is atom)
\+ negates the unification so it fails

and

```
can_lose(X),\+ X=rock.
```

Unification succeeds which will instantiate X with rock

○ Both queries succeed.

○ Both queries fail.

○ The first query succeeds and the second query fails.

○ [✓ The first query fails and the second query succeeds.]

THE UNIVERSITY OF
MELBOURNE

Thank you

wendy.zeng@unimelb.edu.au

By Wendy Zeng